PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-005680
(43)Date of publication of application : 12.01.2001

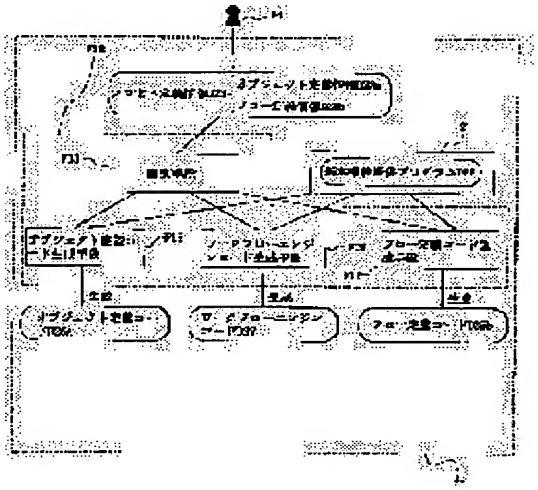
(51)Int.Cl. G06F 9/46
G06F 17/60
G06F 19/00

(21)Application number : 11-179520
(22)Date of filing : 25.06.1999

(71)Applicant : OKI ELECTRIC IND CO LTD
(72)Inventor : OTOGAWA SHINICHI

(54) WORK FLOW MANAGEMENT SYSTEM, CODE GENERATOR, AND METHOD FOR CHANGING PROCESS OF WORK FLOW PROCESSING

(57)Abstract:
PROBLEM TO BE SOLVED: To easily correct the process of work flow processing.
SOLUTION: The process of work flow processing consisting of a series of plural activity elements is defined by inputting process definition data. In the workflow management system 11, a discrimination means F13 separates process definition information D23 inputted by a slip-drafting person 24 into object definition information D23a concerned with the processing contents of each activity and flow definition information D23b concerned with the order of activity elements. An object definition code D25a and a flow definition code D25b are generated from definition information D23a, D23b, respectively. Since definition codes D25a, D25b are independent data of each other, the work flow processing processes to be driven on the basis of these definition codes D25a, D25b can be changed by correcting individual definition codes D25a, D25b.



LEGAL STATUS

[Date of request for examination] 14.02.2003

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2001－5680
（P2001－5680A）
(43)公開日　平成13年１月12日(2001.1.12)

(51)Int.Cl.'	識別記号	ＦＩ		テーマコード'(参考)
Ｇ０６Ｆ　9/46	３６０	Ｇ０６Ｆ　9/46	３６０Ｂ	５Ｂ０４９
17/60		15/21	Ｌ	５Ｂ０９８
19/00		15/22	Ｎ	

審査請求　未請求　請求項の数７　ＯＬ　（全 23 頁）

(21)出願番号　特願平11－179520

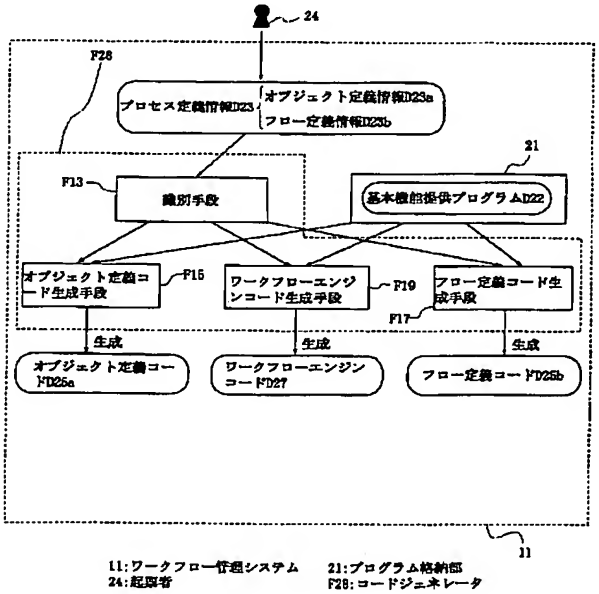
(22)出願日　平成11年６月25日(1999.6.25)

(71)出願人　000000295
沖電気工業株式会社
東京都港区虎ノ門１丁目７番12号
(72)発明者　乙川　進一
東京都港区虎ノ門１丁目７番12号　沖電気工業株式会社内
(74)代理人　100085419
弁理士　大垣　孝

Ｆターム(参考)　5B049 AA01 CC32 DD01 EE31 GG07
5B098 AA10 GA01 GC01 GC16

(54)【発明の名称】　ワークフロー管理システム、コードジェネレータおよびワークフロー処理のプロセス変更方法

(57)【要約】
【課題】　ワークフロー処理のプロセスを容易に修正する。
【解決手段】　プロセス定義データを入力することにより、一連かつ複数のアクティビティからなるワークフロー処理のプロセスは定義される。このワークフロー管理システム１１では、起票者２４によって入力されたプロセス定義情報Ｄ２３は、識別手段Ｆ１３によって、各アクティビティにおける処理内容に関するオブジェクト定義情報Ｄ２３ａおよびアクティビティの順序に関するフロー定義情報Ｄ２３ｂに分離される。オブジェクト定義情報およびフロー定義情報からそれぞれオブジェクト定義コードＤ２５ａおよびフロー定義コードＤ２５ｂを生成する。オブジェクト定義コードおよびフロー定義コードは独立したデータであるため、これら定義コードに基づいて動作するワークフロー処理についてのプロセスの変更は、個々の定義コードの修正により行うことができる。



11:ワークフロー管理システム　21:プログラム格納部
24:起票者　F28:コードジェネレータ

第１の実施の形態のワークフロー管理システム

【特許請求の範囲】
【請求項１】　複数の計算機を有するネットワークシステムで、一連かつ複数の工程からなるワークフロー処理を、各工程における処理内容およびそれら工程の順序に関するプロセス定義情報に基づいて、各工程毎に行うワークフロー管理システムにおいて、
前記プロセス定義情報から、各工程における処理内容に関するオブジェクト定義情報と、工程の順序に関するフロー定義情報とを識別する識別手段と、
前記ワークフロー処理に用いる基本的な機能を提供する複数のプログラムを格納してあるプログラム格納部と、
識別された前記オブジェクト定義情報を解釈して該解釈に応じた前記プログラムを前記プログラム格納部から読み出すことにより、該読み出されたプログラムに基づいて各工程における処理内容に関するコードであるオブジェクト定義コードを生成するオブジェクト定義コード生成手段と、
前記フロー定義情報を解釈して該解釈に応じた前記プログラムを前記プログラム格納部から読み出すことにより、該読み出されたプログラムに基づいて各工程の順序に関するコードであるフロー定義コードを生成するフロー定義コード生成手段と、
前記フロー定義コードおよび前記オブジェクト定義コードを解釈することにより、前記ワークフロー処理を制御するワークフローエンジンとを具えることを特徴とするワークフロー管理システム。
【請求項２】　請求項１に記載のワークフロー管理システムにおいて、
識別された前記オブジェクト定義情報および前記フロー定義情報を解釈して該解釈に応じた前記プログラムを前記プログラム格納部から読み出すことにより、該読み出されたプログラムに基づいてワークフローエンジンコードを生成するワークフローエンジンコード生成手段を更に具え、および、前記ワークフローエンジンコードによって前記ワークフローエンジンを機能させることを特徴とするワークフロー管理システム。
【請求項３】　請求項１に記載のワークフロー管理システムにおいて、
前記オブジェクト定義コードおよび前記フロー定義コードを高水準言語のコードの状態で保存する定義コード保存手段を更に具えることを特徴とするワークフロー管理システム。
【請求項４】　請求項１に記載のワークフロー管理システムにおいて、
前記ワークフローエンジンが上位階層から複数の下位階層に機能を継承する階層構造を有し、かつ、個々の前記下位階層のワークフローエンジンは相互にアクセスする通信機能を有することを特徴とするワークフロー管理システム。
【請求項５】　請求項４に記載のワークフロー管理システムにおいて、
前記フロー定義コードに一般的な宛先名を予め定義する場合、該一般的宛先名から具体的宛先名を生成する宛名変換手段を更に具えており、および、
前記宛名変換手段が、前記下位階層のワークフローエンジンと相互にアクセスする通信機能を有することを特徴とするワークフロー管理システム。
【請求項６】　複数の計算機を有するネットワークシステムで、一連かつ複数の工程からなるワークフロー処理を行う際、各工程における処理内容およびそれら工程の順序に関するプロセス定義情報から、ワークフローエンジンの解釈できるプロセスに関するコードを生成するコードジェネレータであって、
前記プロセス定義情報から、各工程における処理内容に関するオブジェクト定義情報と、工程の順序に関するフロー定義情報とを識別する識別手段と、
前記オブジェクト定義情報から、ワークフローエンジンの解釈できるオブジェクト定義コードを生成するオブジェクト定義コード生成手段と、
前記フロー定義情報から、ワークフローエンジンの解釈できるフロー定義コードを生成するフロー定義コード生成手段とを具えることを特徴とするコードジェネレータ。
【請求項７】　一連かつ複数の工程からなるワークフロー処理を行う前に、各工程の処理内容および工程の処理順序に関するプロセス定義情報から、各工程の処理内容に関するオブジェクト定義情報と、工程の処理順序に関するフロー定義情報とを識別し、前記オブジェクト定義情報からワークフローエンジンの解釈できるオブジェクト定義コードを生成し、かつ、前記フロー定義情報からワークフローエンジンの解釈できるフロー定義コードを生成するワークフロー処理のプロセスを変更するに当たり、
各工程の処理内容のみを変更する場合にはオブジェクト定義コードのみを修正し、工程の処理順序のみを変更する場合にはフロー定義コードのみを修正することを特徴とするワークフロー処理のプロセス変更方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】この発明は、ワークフロー管理システム、ワークフロー管理システムに用いるコードジェネレータおよびワークフロー処理のプロセスの変更方法に関する。
【０００２】
【従来の技術】従来、ワークフロー管理システムが、複数の端末計算機が接続されたネットワークシステムで、一連かつ複数の工程からなる情報処理すなわちワークフロー処理を効率よく行うために利用されている。ワークフロー処理に係るデータは、パッケージとして各計算機の間を定められた順序で移動し、および、各計算機で処

理される。これらの各計算機を操作して各処理者がデータを参照したり書き換えたりする。このような、ワークフロー管理システムの一例が文献Ｉ（日経コンピュータ」１９９７．９．１号　第２０４～２１７頁）に開示されている。

【０００３】図９は、文献Ｉに開示されているワークフロー管理システムを各機能毎にブロックで分割して概略的に示す図である。図９では、データを角の丸い長方形で示し、機能手段（アプリケーション等）を角張った長方形で示してある。図９に示すように、このワークフロー管理システム２０１は、ワークフローエンジンＡ２０３と、プロセス定義ツールＡ２０５と、ワークリストハンドラＡ２０７と、モニタリングツールＡ２０９とを有する。広く用いられているクライアント／サーバシステムでは、ワークフロー処理を制御するワークフローエンジンＡ２０３がワークフローサーバ２０４で動作しており、各処理者はクライアント計算機を操作して担当の工程（アクティビティ）を実行する。

【０００４】プロセス定義ツールＡ２０５は、各工程の処理内容と、アクティビティの順序とを定義するための機能モジュールである。プロセス定義ツールＡ２０５は、新規にワークフロー処理を開始するとき、プロセス定義データＤ２１１を生成する。プロセス定義データＤ２１１は、各アクティビティにおける処理内容と、アクティビティの順序とに関するデータを一体のデータとして含む。また、プロセス定義データＤ２１１は、ワークフロー処理の終了条件等の制御条件に関するデータを含むことがある。一般的なプロセス定義ツールＡ２０５ではグラフィカルユーザインターフェース（ＧＵＩ）環境が整えられている。そのため、処理者は、マウス操作によってポインタを用いることによりプロセス（アクティビティの順序および各アクティビティでの処理内容）を容易に定義できる。

【０００５】ワークフローエンジンＡ２０３は、ワークフロー処理を制御する機能モジュールである。具体的に言うと、ワークフローエンジンＡ２０３は、プロセス定義データＤ２１１を解釈してこの解釈に基づいて処理内容を示すワークアイテムをワークリストＤ２１３に追加する。なお、ワークリストＤ２１３は、各処理者が処理すべきワークアイテムを格納するデータ領域であり、個々の処理者毎に設けられる。

【０００６】各処理者は、ワークリストハンドラＡ２０７によって示されるワークアイテムにしたがって、クライアントアプリケーションＡ２１５等を用いてアクティビティを実行する。また、人間による処理を必要としないアクティビティについては、ワークフローエンジンＡ２０３によって起動された起動アプリケーションＡ２１７が、アクティビティを実行する。図９に示すように、これらクライアントアプリケーションＡ２１５或いは起動アプリケーションＡ２１７はアプリケーションデータＤ２１９を利用することがある。

【０００７】ワークリストハンドラＡ２０７は、ワークリストＤ２１３を参照して処理すべきワークアイテムを各処理者に示すことにより、各処理者に処理内容に関する指示を与える。その処理者が作業を終了すると、ワークリストハンドラＡ２０７はワークリストＤ２１３からワークアイテムを削除する。ワークフローエンジンＡ２０３は、ワークアイテムの削除によってアクティビティの終了を認識して、次の処理者のワークリストＤ２１３にワークアイテムを付加する。

【０００８】モニタリングツールＡ２０９は、ワークフローエンジンＡ２０３を監視することによってワークフロー処理の進行具合を確認する機能モジュールである。モニタリングツールＡ２０９によって、監督者或いは各処理者は必要に応じてワークフロー処理の途中経過や終了結果を参照することができる。

【０００９】ワークフロー制御データＤ２２１は、複数のワークフロー処理を並行して行う場合、各ワークフロー処理を受け持つワークフローエンジンを互いに識別するのに用いられる。また、ワークフロー関連データＤ２２３は、ワークフロー処理に関連する様々な情報を含む。ワークフロー関連データＤ２２３は、例えば購入物品の価格データ等であり、価格帯によって担当すべき処理者が変化する場合等に用いられる。

【００１０】図９に示すように、ワークフローエンジンＡ２０３がプロセス定義データＤ２１１に基づいて各アクティビティにおける処理を制御することにより、ワークフロー処理は行われる。一般的に言うと、ワークフロー処理は、繰り返し行われる定型業務およびアドホックな非定型業務の二つの業務に分類される。定型業務には出張費伝票、備品購入伝票または予算申請などがあり、非定型業務には資料回覧、技術文書作成またはソフト開発などがある。業務の効率を向上させるためには、これら多種多様な業務に適用できるワークフロー管理システムが必要である。

【００１１】典型的に言うと、定型業務のワークフロー処理では、アクティビティの順序および各アクティビティにおける処理内容が殆ど変化しないため、一つのプロセス定義データを用いることが多い。一方、非定型業務のワークフロー処理では、アクティビティの順序および各アクティビティにおける処理内容が一定ではないので、ワークフロー処理を行う毎にプロセス定義データを生成することが多い。

【００１２】

【発明が解決しようとする課題】しかしながら、従来のワークフロー管理システムでは、定型業務或いは非定型業務のいずれにおいても、各アクティビティの処理内容或いは処理順序を変更する場合にはプロセス定義データの全体を修正する必要があった。例えば、ルーチンワークである備品購入伝票において決裁者を変更する場合に

は、今まで用いていたプロセス定義データを修正しなければならない。また、アドホックに発生する資料回覧において資料の回覧先を変更したい場合には、資料回覧を開始する前にプロセス定義データを修正しなければならない。

【００１３】このようなアクティビティの処理順序の変更を容易に行うため、図９のワークフロー管理システム２０１では、ワークフローエンジンＡ２０３が組織／役割データＤ２２５を参照する。組織／役割データＤ２２５とは、課長や部長などの組織内の役職等を示す一般的宛先名と、処理者の名前等の具体的宛先名とを関連付けたデータである。この場合、プロセス定義データＤ２１１に一般的宛先名を定義しておくだけで、ワークフローエンジンＡ２０３は、組織／役割データＤ２２５を参照することにより一般的宛先名から具体的宛先名を判断できる。そのため、決裁ルートに沿う役職にある処理者が人事異動等によって他の処理者に変更された場合でも、動的に宛先が認識されるため、決裁ルートの宛先名を変更する必要が無くなる。

【００１４】このような組織／役割データの導入によって変更が不必要になる場合はあるが、役職で処理順序を定義しないワークフロー処理では、依然としてアクティビティの処理順序をプロセス定義データから修正する必要があった。

【００１５】したがって、定型業務および非定型業務のいずれのワークフロー処理においても、一つのアクティビティにおける処理内容を変更する場合、或いは、アクティビティの処理順序を変更する場合には、プロセス定義データを修正し或いはプロセス定義データを新たに作成する。このように、従来のワークフロー処理では、プロセスに関する変更はプロセス定義データの再定義を伴うため非能率的であり、容易に行うことができなかった（第１の問題点）。

【００１６】また、通常、新たにワークフロー処理を開始するときには、プロセス定義データを新規に作成する。このとき、ワークフロー処理を適用する業務の種類が多岐に渡る場合には、必然的にプロセス定義データも様々な内容を有することとなる。ワークフローエンジンは、それらのプロセス定義データを解釈してかつ実行できなければならないから、多種のワークフロー処理を実行するための多くの基本的な機能を予め具備している必要がある。したがって、ワークフローエンジンの構成が複雑となり、ワークフローエンジンとして機能するワークフローエンジンコードの占有メモリが必然的に大きくなる（第２の問題点）。

【００１７】そのため、少なくとも第１の問題点を解決でき、好ましくは第２の問題点も解決できるワークフロー管理システムが望まれていた。

【００１８】

【課題を解決するための手段】したがって、この出願に係る発明のワークフロー管理システムは、複数の計算機を有するネットワークシステムで、一連かつ複数の工程からなるワークフロー処理を、各工程における処理内容およびそれら工程の順序に関するプロセス定義情報に基づいて、各工程ごとに行うワークフロー管理システムにおいて、前述のプロセス定義情報から、各工程における処理内容に関するオブジェクト定義情報と、工程の順序に関するフロー定義情報とを識別する識別手段と、前述のワークフロー処理に用いる基本的な機能を提供する複数のプログラムを格納してあるプログラム格納部と、前述のオブジェクト定義情報を解釈してこの解釈に応じたプログラムをプログラム格納部から読み出し、かつ、この読み出されたプログラムに基づいて各工程における処理内容に関するコードであるオブジェクト定義コードを生成するオブジェクト定義コード生成手段と、前述のフロー定義情報を解釈してこの解釈に応じたプログラムをプログラム格納部から読み出し、かつ、この読み出された各工程の順序に関するコードであるフロー定義コードを生成するフロー定義コード生成手段と、前述のフロー定義コードおよび前述のオブジェクト定義コードを解釈することにより、ワークフロー処理を制御するワークフローエンジンとを具える。

【００１９】このワークフロー管理システムの構成によれば、ワークフロー処理のプロセスを定義をする際、起票者がワークフロー処理を開始させるためにプロセスを定義する。起票者が入力したプロセス定義情報は、識別手段によってフロー定義に関する部分およびオブジェクト定義に関する部分に分離される。この分離の後、前者の部分はフロー定義コード生成手段によってコード化されてフロー定義コードとして生成され、および、後者の部分はオブジェクト定義コード生成手段によってコード化されてオブジェクト定義コードとして生成される。よって、ワークフロー処理を行うとき、プロセス（各アクティビティにおける処理内容およびアクティビティの順序）を変更する場合であっても、プロセス定義データそのものを修正する必要がない。言い換えれば、各アクティビティにおける処理内容のみを変更したい場合にはオブジェクト定義コードのみを修正すれば良く、アクティビティの順序（アクティビティを行う処理者の変更）を変更したい場合にはフロー定義コードのみを修正すればよい。

【００２０】また、この発明の実施に当たり、より好適には、前述のオブジェクト定義情報および前述のフロー定義情報を解釈してこの解釈に応じたプログラムをプログラム格納部から読み出すことにより、この読み出されたプログラムに基づいてワークフローエンジンコードを生成するワークフローエンジンコード生成手段を更に具え、および、このワークフローエンジンコードによって前述のワークフローエンジンコードを機能させるのが良い。

【００２１】それにより、フロー定義情報およびオブジェクト定義情報に定義された機能を実現するための必要最低限の構成のみを有するワークフローエンジンコードを生成できる。よって、ワークフローエンジンコードの占有メモリを小さくできる。なお、ワークフローエンジンコードとは、ＣＰＵで実行されることにより仮想的にＣＰＵにワークフローエンジンを存在させかつ機能させるプログラムコードを意味する。

【００２２】また、この発明の実施に当たり、好適には、前述のオブジェクト定義コードおよび前述のフロー定義コードを高水準言語のコードの状態で保存する定義コード保存手段を更に具えるのが良い。それにより、フロー定義或いはオブジェクト定義についての修正をするとき、修正する者は、各定義コードを理解しながらコードを追加したり削除したり或いは修正したりすることができる。

【００２３】また、この発明の実施に当たり、好適には、前述のワークフローエンジンは上位階層から複数の下位階層に機能を継承する階層構造を有し、かつ、個々の下位階層のワークフローエンジンは相互にアクセスする通信機能を有するのがよい。それにより、下位階層のワークフローエンジンをネットワークシステム内のホスト計算機に分散できる。このように、ワークフローエンジンに階層構造を与え、個々の下位階層のワークフローエンジンに通信機能を設けると、下位下層のワークフローエンジンは、ネットワークシステム内の任意のホスト計算機に分散して存在できる。よって、負荷分散によるネットワークシステムの効率および安定性が実現できる。

【００２４】また、上述のように下位階層のワークフローエンジンを分散させてなるワークフロー管理システムでは、前述のフロー定義コードに一般的な宛先名を予め定義する場合、一般的宛先名から具体的宛先名を生成する宛名変換手段を更に具えていて、この宛名変換手段が、ワークフローエンジンと相互にアクセスする通信機能を有するのが望ましい。

【００２５】従来、宛名変換を行うには、宛名変換手段はワークフローエンジンの存在するホスト計算機と同一のホスト計算機に設ける必要があった。しかしながら、宛名変換手段に通信機能を設けたため、上述のように下位階層のワークフローエンジンを分散させた場合であっても、宛名変換手段は下位階層のワークフローエンジンと相互にアクセスできる。よって、宛名変換手段を下位階層のワークフローエンジンと別のホスト計算機に設けることができる。

【００２６】また、この出願に係る発明のコードジェネレータは、複数の計算機を有するネットワークシステムで、一連かつ複数の工程からなるワークフロー処理を行う際、各工程における処理内容およびそれら工程の順序に関するプロセス定義情報から、ワークフローエンジンの解釈できるコードを生成するコードジェネレータであって、前述のプロセス定義情報から、各工程における処理内容に関するオブジェクト定義情報と、工程の順序に関するフロー定義情報とを識別する識別手段と、前述のオブジェクト定義情報からワークフローエンジンの解釈できるオブジェクト定義コードを生成するオブジェクト定義コード生成手段と、前述のフロー定義情報からワークフローエンジンの解釈できるフロー定義コードを生成するフロー定義コード生成手段とを具える。

【００２７】このコードジェネレータの構成によれば、識別手段が、プロセス定義情報をオブジェクト定義情報およびフロー定義情報に分離させたのち、オブジェクト定義情報からオブジェクト定義コードを生成しかつフロー定義情報からフロー定義コードを生成する。そのため、各工程における処理内容のみを変更したい場合にはオブジェクト定義コードを修正し、また、工程の処理順序のみを変更したい場合にはフロー定義コードを修正すればよい。よって、プロセスを能率的にかつ容易に変更することができる。なお、オブジェクト定義コードおよびフロー定義コードは、例えば、高水準言語としてのＪａｖａ言語或いはC++のソースプログラムコードすなわち、オブジェクト定義ソースプログラムコードおよびフロー定義ソースプログラムコードとして生成される。このとき、これらオブジェクト定義コードおよびフロー定義コードは、コンパイルされると、ワークフローエンジンが解釈できる実行型プログラムコードとなる。

【００２８】また、この出願に係るワークフロー処理のプロセスの変更方法の発明では、一連かつ複数の工程からなるワークフロー処理を行う前に、各工程の処理内容および工程の処理順序に関するプロセス定義情報から、各工程の処理内容に関するオブジェクト定義情報と、工程の処理順序に関するフロー定義情報とを識別し、オブジェクト定義情報からワークフローエンジンの解釈できるオブジェクト定義コードを生成し、かつ、フロー定義情報からワークフローエンジンの解釈できるフロー定義コードを生成するワークフロー処理について、当該ワークフロー処理のプロセスを変更するに当たり、各工程の処理内容のみを変更する場合にはオブジェクト定義コードのみを修正し、工程の処理順序のみを変更する場合にはフロー定義コードのみを修正する。

【００２９】このプロセスの変更方法によれば、各工程における処理内容のみを変更したい場合若しくは工程の処理順序のみを変更したい場合のいずれにも、互いに独立に存在するオブジェクト定義コード或いはフロー定義コードの一方を修正することにより対応できる。よって、プロセスを能率的にかつ容易に変更できる。

【００３０】

【発明の実施の形態】以下、図を参照して、この発明のワークフロー管理システムの実施の形態につき説明する。なお、この説明に用いる各ブロック図は、これら発

明を理解できる程度に各構成成分をブロックで表して概略的な機能上の相互関係を示しているに過ぎない。また、各図において同様な構成成分については、同一の番号を付して示し、その重複する説明を省略することがある。また、各図で角張った長方形は機能単位を示し、角の丸い長方形はデータを示す。

【００３１】（第１の実施の形態）ワークフロー管理システムでは、ワークフロー処理のプロセスを定義したのち当該ワークフロー処理を実行する必要がある。そのため、第１の実施の形態では、ワークフロー管理システムのプロセスを定義する形態につき説明し、次にワークフロー管理システムのワークフロー処理を実行する形態につき説明し、続いてワークフロー処理のプロセスを変更する方法につき説明する。

【００３２】図１は、第１の実施の形態のワークフロー管理システムについて、主要な機能単位の相互関係を概略的に示すブロック図である。ただし、図１中の各機能単位は、ハードウェア資源としての中央処理装置がその機能単位に対応するプログラムを実行することにより、仮想的に中央処理装置に存在する。以下では、中央処理装置に仮想的に実現される機能単位を角張った長方形、記憶装置のデータ領域に形成されるデータ情報を丸みを帯びた長方形で示してある。また、図２は、図１の各機能単位を１台のホスト計算機で動作させた場合を例として、各機能単位によるハードウェア資源の利用形態を概略的に示す図である。

【００３３】以下、図１および図２を参照して、実施の形態のワークフロー管理システムのプロセスを定義する形態につき説明する。ただし、実施の形態のワークフロー管理システムでは特にワークフローエンジンコード生成手段が設けてある。

【００３４】周知のごとく、ワークフロー管理システムは、複数のホスト計算機を有するネットワークシステムで、一連かつ複数のアクティビティからなるワークフロー処理を各アクティビティ毎に行うことにより、ワークフロー処理を実行する。

【００３５】図１に示す第１の実施の形態のワークフロー管理システム１１は、識別手段Ｆ１３と、オブジェクト定義コード生成手段Ｆ１５と、フロー定義コード生成手段Ｆ１７と、ワークフローエンジンコード生成手段Ｆ１９と、プログラム格納部２１とを有する。なお、ここではこれら各機能単位Ｆ１３、Ｆ１５、Ｆ１７およびＦ１９は、図２のホスト計算機ｈ１の中央処理装置（ＣＰＵ）ｃ１で仮想的な機能手段として動作するように構成されている。

【００３６】図２に示すホスト計算機ｈ１は、ハードディスクや光ディスク等の記憶装置ｍ１と、ＣＰＵｃ１と、マンマシンインターフェース用の入力装置ｉ１とを有する。ＣＰＵｃ１には、これら記憶装置ｍ１、入力装置ｉ１、上述の各機能単位Ｆ１３、Ｆ１５、Ｆ１７およびＦ１９その他の動作のタイミングや始動或いは停止を含む所要の制御を行うための制御部を図示せずも含んでいる。なお、ホスト計算機ｈ１の記憶装置ｍ１には、識別手段Ｆ１３の機能を実現させる識別プログラムＤ１３と、オブジェクト定義コード生成手段Ｆ１５の機能を実現させるオブジェクト定義コード生成プログラムＤ１５と、フロー定義コード生成手段Ｆ１７の機能を実現させるフロー定義コード生成プログラムＤ１７と、ワークフローエンジンコード生成手段Ｆ１９の機能を実現させるワークフローエンジンコード生成プログラムＤ１９とが格納されている。

【００３７】先ず、ワークフロー処理のプロセスを定義するに当たり、起票者２４は、ホスト計算機ｈ１の入力装置ｉ１を操作して記憶装置ｍ１にプロセス定義情報Ｄ２３を格納する。このとき、起票者２４は、ホスト計算機ｈ１の入力装置ｉ１から例えばキーボードからのスクリプト入力或いはポインタデバイスからのＧＵＩ操作によって、プロセス定義情報Ｄ２３を入力する。これにより、プロセス定義情報Ｄ２３がホスト計算機ｈ１の記憶装置ｍ１に格納される。ただし、プロセス定義情報Ｄ２３は、各アクティビティの処理内容に関するデータすなわちオブジェクト定義情報Ｄ２３ａと、アクティビティの順序に関するデータすなわちフロー定義情報Ｄ２３ｂとを含む。

【００３８】プロセス定義情報Ｄ２３が入力されると、識別手段Ｆ１３は、記憶装置ｍ１に格納されたプロセス定義情報Ｄ２３を読み出してきて、このプロセス定義情報Ｄ２３からオブジェクト定義情報Ｄ２３ａおよびフロー定義情報Ｄ２３ｂを互いに識別する。しかる後、識別されたオブジェクト定義情報Ｄ２３ａおよびフロー定義情報Ｄ２３ｂを記憶装置ｍ１に格納する。なお、識別手段Ｆ１３は、記憶装置ｍ１に格納されている識別プログラムＤ１３をＣＰＵｃ１が読み出しかつ実行することにより、仮想的な機能単位としてＣＰＵｃ１で動作する。

【００３９】プログラム格納部２１は、ワークフロー処理に用いる基本的な機能を提供する複数の基本機能提供プログラムＤ２２を格納している。なお、プログラム格納部２１は、例えばホスト計算機ｈ１の記憶装置ｍ１内のデータ領域に形成される。

【００４０】例えば、この基本機能提供プログラムＤ２２は、ワークフロー処理に用いる基本的な機能を各機能毎に区分された複数のプログラムからなる。このように、基本的な機能が各機能毎に区分されていると、不必要な機能を読み出すことなく必要な機能だけを読み出すことができる。したがって、基本機能提供プログラムＤ２２から生成されるオブジェクト定義コードＤ２５ａ、フロー定義コードＤ２５ｂおよびワークフローエンジンコードＤ２７の占有メモリを小さくできる。

【００４１】オブジェクト定義コード生成手段Ｆ１５は、オブジェクト定義情報Ｄ２３ａを解釈してこの解釈

に応じた基本機能提供プログラムＤ２２をプログラム格納部２１から読み出すことにより、読み出された基本機能提供プログラムＤ２２に基づいてオブジェクト定義コードＤ２５ａを生成する。このとき、具体的にはオブジェクト定義コード生成手段Ｆ１５はホスト計算機ｈ１内で次のように機能する。

【００４２】すなわち、ホスト計算機ｈ１のＣＰＵｃ１が記憶装置ｍ１に格納されているオブジェクト定義コード生成プログラムＤ１５を読み出してかつ実行する。それにより、オブジェクト定義コード生成手段Ｆ１５が起動する。オブジェクト定義コード生成手段Ｆ１５は、記憶装置ｍ１からオブジェクト定義情報Ｄ２３ａを読み出すと共に、オブジェクト定義情報Ｄ２３ａに基づいてオブジェクト定義コードＤ２５ａの生成に必要な基本機能提供プログラムＤ２２かどうかを判断して当該基本機能提供プログラムＤ２２を読み出す。しかるのち、オブジェクト定義コード生成手段Ｆ１５は、読み出された基本機能提供プログラムＤ２２およびオブジェクト定義情報Ｄ２３ａを例えば組み合わせてオブジェクト定義コードＤ２５ａを生成する。例えば、このときオブジェクト定義コード生成手段Ｆ１５は、複数の基本機能提供プログラムＤ２２を読み出してオブジェクト定義コードＤ２５ａを生成しても良い。なお、生成されたオブジェクト定義コードＤ２５ａは、少なくとも各アクティビティにおける共通の処理内容を定義した基本ソースプログラムを含むソースプログラムとして記憶装置ｍ１に格納される。

【００４３】フロー定義コード生成手段Ｆ１７は、フロー定義情報Ｄ２３ｂを解釈してこの解釈に応じた基本機能提供プログラムＤ２２をプログラム格納手段２１から読み出すことにより、読み出された基本機能提供プログラムＤ２２に基づいてフロー定義コードＤ２５ｂを生成する。このとき、具体的にはフロー定義コード生成手段Ｆ１７はホスト計算機ｈ１内で次のように機能する。

【００４４】すなわち、ホスト計算機ｈ１のＣＰＵｃ１が記憶装置ｍ１に格納されているフロー定義コード生成プログラムＤ１７を読み出してかつ実行する。それにより、フロー定義コード生成手段Ｆ１７が起動する。フロー定義コード生成手段Ｆ１７は、記憶装置ｍ１からフロー定義情報Ｄ２３ｂを読み出すと共に、フロー定義情報Ｄ２３ｂに基づいてフロー定義コードＤ２５ｂの生成に必要な基本機能提供プログラムＤ２２かどうかを判断して当該基本機能提供プログラムＤ２２を読み出す。しかるのち、フロー定義コード生成手段Ｆ１７は、読み出された基本機能提供プログラムＤ２２およびフロー定義情報Ｄ２３ｂを例えば組み合わせてフロー定義コードＤ２５ｂを生成する。例えば、このときフロー定義コード生成手段Ｆ１７は、複数の基本機能提供プログラムＤ２２を読み出してフロー定義コードＤ２５ｂを生成しても良い。なお、生成されたフロー定義コードＤ２５ｂは、少なくともアクティビティの順序を定義したソースプログラムを含むソースプログラムとして記憶装置ｍ１に格納される。

【００４５】ワークフローエンジンコード生成手段Ｆ１９は、識別されたオブジェクト定義情報Ｄ２３ａおよびフロー定義情報Ｄ２３ｂを解釈してこの解釈に応じた基本機能提供プログラムＤ２２をプログラム格納部２１から読み出すことにより、読み出された基本機能提供プログラムＤ２２に基づいてワークフローエンジンコードＤ２７を生成する。このとき、具体的にはワークフローエンジンコード生成手段Ｆ１９はホスト計算機ｈ１内で次のように機能する。

【００４６】すなわち、ホスト計算機ｈ１のＣＰＵｃ１が記憶装置ｍ１に格納されているワークフローエンジンコード生成プログラムＤ１９を読み出してかつ実行することにより、ワークフローエンジンコード生成手段Ｆ１９が起動する。起動したワークフローエンジンコード生成手段Ｆ１９は、記憶装置ｍ１からオブジェクト定義情報Ｄ２３ａおよびフロー定義情報Ｄ２３ｂを読み出すと共に、これらの定義情報Ｄ２３ａおよびＤ２３ｂに基づいてワークフローエンジンコードＤ２７の生成に必要な基本機能提供プログラムＤ２２かどうかを判断して当該基本機能提供プログラムＤ２２を読み出す。しかるのち、ワークフローエンジンコード生成手段Ｆ１９は、読み出された基本機能提供プログラムＤ２２からワークフローエンジンコードＤ２７を生成する。例えば、このときワークフローエンジンコード生成手段Ｆ１９は、複数の基本機能提供プログラムＤ２２を組み合わせてワークフローエンジンコードＤ２７を生成する。なお、生成されたワークフローエンジンコードＤ２７は、オブジェクト定義コードＤ１５およびフロー定義コードＤ１７を解釈（ただし、Ｊａｖａプログラミングにおける「参照」の意味を含む。）してかつこの解釈に基づいてワークフロー処理を制御するためのソースプログラムとして記憶装置ｍ１に格納される。

【００４７】また、実際にワークフロー処理を行う際には、例えばホスト計算機ｈ１のＣＰＵｃ１が記憶装置ｍ１に格納されているワークフローエンジンコードＤ２７を読み出してかつ実行することにより、一つの機能単位としてのワークフローエンジンが起動してワークフロー処理の制御動作を行う。

【００４８】なお、識別手段Ｆ１３、オブジェクト定義コード生成手段Ｆ１５およびフロー定義コード生成手段Ｆ１７は、コードジェネレータＦ２８という一つの機能単位として認識できる。コードジェネレータＦ２８は、複数のホスト計算機を有するネットワークシステムで、一連かつ複数のアクティビティからなるワークフロー処理を行う際、各アクティビティにおける処理内容およびそれらアクティビティの順序に関するプロセス定義情報から、ワークフローエンジンの解釈できるプロセスに関

するコードを生成する。特に、このコードジェネレータＦ２８は、プロセス定義情報Ｄ２３から、オブジェクト定義コードＤ２５ａおよびフロー定義コードＤ２５ｂを区別してそれぞれ生成する。そのため、一般的なワークフロー管理システムについて、そのワークフローエンジンがオブジェクト定義コードＤ２５ａおよびフロー定義コードＤ２５ｂを解釈できる構成であれば、このコードジェネレータＦ２８の部分のみを適用できる。

【００４９】ここで、具体的なワークフロー処理を実行する場合を例として、以上説明した各機能単位につき更に詳細に説明する。

【００５０】なお、ここではワークフロー管理システムを、オブジェクト指向プログラミング（ＯＯＰ）を用いて構築する。周知の如く、Ｊａｖａ言語やＣ＋＋言語等は、ＯＯＰに適している。ＯＯＰを用いると、上述したワークフローエンジンについて、基本的な機能のみを有する上位階層（スーパークラス）から具体的かつ個別な機能を有する下位階層（サブクラス）までを含む、階層構造を有するワークフローエンジンとして定義できる。このとき、オブジェクト定義コードおよびフロー定義コードは、ワークフローエンジンに参照されるオブジェクトとなる。

【００５１】図３は、図１のワークフロー管理システムの具体的な機能構成を概略的に示した図であって、図１に示すワークフローエンジンに代えてクラス構造すなわち階層構造を有するワークフローエンジンを用いる場合のワークフロー管理システムについて、その主要な機能単位の相互関係を概略的に示すブロック図である。なお、図３の各機能単位は、ハードウェア資源としてのＣＰＵがその機能単位に対応するプログラムを実行することにより、仮想的にＣＰＵに存在する機能単位である。ただし、便宜上、図３ではワークフローエンジンサブクラスＦ２７ａおよびＦ２７ｂは、それぞれホスト計算機ｈ２およびｈ３の各ＣＰＵで動作する仮想的な機能単位として示してある。

【００５２】図４は、図３のワークフロー管理システムの各機能単位を動作させるネットワークシステムの一例を示す図である。図４には３つのホスト計算機ｈ１、ｈ２およびｈ３が接続されたネットワークシステムが示されている。図４に示すように、各ホスト計算機ｈ１、ｈ２およびｈ３は、ＣＰＵｃ１、ｃ２およびｃ３と、ハードディスクや光ディスク等の記憶装置ｍ１、ｍ２およびｍ３と、マンマシンインターフェース用の入力装置ｉ１、ｉ２およびｉ３とを具える。また、図示せずも、各ホスト計算機ｈ１、ｈ２およびｈ３は一般に表示装置等を有する。既に説明したようにＣＰＵｃ１、ｃ２およびｃ３は、各機能単位を駆動するための制御部を有している。また、ＣＰＵｃ１、ｃ２およびｃ３には各ハードウェア資源を駆動するドライバソフトや一般的なＯＳソフト等が動作していても良い。なお、図４は、ワークフロー管理システムの各構成が存在できる計算機の一例を示しているに過ぎない。

【００５３】ここで具体的なワークフロー処理の例として、２つのアクティビティからなるワークフロー処理を実施する。すなわち、図３および図４に示すように、このワークフロー処理では、先ず起票者２４がホスト計算機ｈ１を操作してワークフロー処理のプロセスを定義し、次に処理者３１ａがホスト計算機ｈ２を操作してアクティビティを実行し、その後、処理者３１ｂがホスト計算機ｈ３を操作してアクティビティを実行する。

【００５４】図３および図４に示すワークフローエンジンは、上位階層から下位階層までの複数の階層構造を有する。すなわち、このワークフローエンジンを機能させるコードは、上位階層のワークフローエンジンコード（ワークフローエンジンスーパークラスコード）Ｄ２７ｓ、下位階層のワークフローエンジンコード（ワークフローエンジンサブクラスコード）Ｄ２７ａおよびＤ２７ｂという２層の階層構造を有している。

【００５５】ただし、ここではワークフローエンジンコード生成手段Ｆ１９は、ソースプログラムコードであるワークフローエンジンサブクラスコードＤ２７ａおよびＤ２７ｂを生成する。なお、このワークフローエンジンサブクラスコードＤ２７ａは、ワークフローエンジンサブクラスＦ２７ａを機能させ、ワークフローエンジンサブクラスコードＤ２７ｂは、ワークフローエンジンサブクラスＦ２７ｂを機能させる。

【００５６】このように階層構造を有するワークフローエンジンを用いる場合、下位階層のワークフローエンジンコードのみを生成するのが好ましい。

【００５７】周知の如く、ワークフローエンジンサブクラスＦ２７ａおよびＦ２７ｂは、ワークフローエンジンスーパークラス（図中では機能単位ではなくコードとして示してある。）Ｄ２７ｓから機能的な継承（インヘリタンス）を行うことができる。したがって、ワークフローエンジンスーパークラスコードＤ２７ｓで基本機能を定義しておけば、ワークフローエンジンサブクラスＦ２７ａおよびＦ２７ｂでは、当該基本機能を継承できる。そのため、ワークフローエンジンサブクラスコードＤ２７ａおよびＤ２７ｂでは、当該基本機能と異なる機能のみを定義すればよい。

【００５８】このように、ワークフローエンジンコード生成手段Ｆ１９は、ＯＯＰに基づいて、基本機能を定義した上位階層のワークフローエンジンコードＤ２７ｓから機能的な継承をする下位階層のワークフローエンジンコードＤ２７ａおよびＤ２７ｂであって、かつ、具体的な固有の機能を定義した下位階層のワークフローエンジンコードＤ２７ａおよびＤ２７ｂを生成するのが好ましい。それにより、ワークフローエンジンとして機能させる全体のソースプログラムコードの記述を簡略化できるため、ワークフローエンジンコードを容易に生成でき

る。それと同時に、ワークフローエンジンコードの記憶装置内の占有メモリを低減できる。なお、図３に示すように、ワークフローエンジンスーパークラスコードＤ２７ｓ等を含むクラスライブラリＤ５２は、例えばプログラム格納部２１に設けておく。

【００５９】また、スーパクラスからサブクラスに渡る階層構造を有するワークフローエンジンコードについて、ワークフローエンジンサブクラスコードＤ２７ａおよびＤ２７ｂに通信機能を設定して互いにアクセスできるようにしておけば、ワークフローエンジンサブクラスＤ２７ａおよびＤ２７ｂは、それぞれネットワークシステム内の任意のホスト計算機に分散して存在できる。よって、負荷分散によるネットワークシステムの効率および安定性の向上が実現できる。

【００６０】なお、ここでは、図４に示すように、ホスト計算機ｈ１にて生成されたワークフローエンジンサブクラスコードＤ２７ａおよびＤ２７ｂは、それぞれホスト計算機ｈ２およびｈ３に分散している。このように、図４のワークフローエンジンサブクラスコードＤ２７ａはホスト計算機ｈ２で動作して図３に示すワークフローエンジンサブクラスＦ２７ａとして機能し、図４のワークフローエンジンサブクラスコードＤ２７ｂはホスト計算機ｈ３で動作して図３に示すワークフローエンジンサブクラスＦ２７ｂとして機能する。図３および図４に示すように、処理者３１ａは、ホスト計算機ｈ２を操作することによりアクティビティを行い。その後、処理者３１ｂは、ホスト計算機ｈ３を操作することにより別のアクティビティを行う。このように、ワークフローエンジンコードを生成するとき、各処理者のホスト計算機毎に、互いに異なるワークフローエンジンサブクラスを定義するのが良い。それにより、容易に処理者の特定ができるため、ワークフロー処理のプロセスを定義することが容易となる。

【００６１】ここでは、上述のごとく処理者毎に下位階層のワークフローエンジンコードを生成するため、次のようなプロセス定義情報を生成する。すなわち、図３では、前述したプロセス定義情報として例えばプロセス定義スクリプトＤ４３を用いる。このとき、起票者２４は例えばホスト計算機ｈ１の入力装置ｉ１を操作してプロセス定義スクリプトＤ４３を入力する。このプロセス定義スクリプトＤ４３は、オブジェクト定義スクリプトＤ４３ａおよびフロー定義スクリプトＤ４３ｂを含む。

【００６２】このとき、ホスト計算機ｈ１のＣＰＵｃ１にＯＳソフトやエディタソフトを動作させておくと、記憶装置ｍ１にはデータとしてのプロセス定義スクリプトＤ４３が格納される。なお、このプロセス定義スクリプトＤ４３は、ワークフロー処理のプロセス定義に関する情報を簡単な内部コードで記述したデータである。ただし、ここでいう内部コードとはスクリプト識別手段Ｆ４５が認識できるコードである。

【００６３】図５は、具体的なワークフロー処理に対するプロセス定義スクリプトの一例である。図５に示すプロセス定義スクリプトは、オブジェクト定義スクリプトＤ４３ａおよびフロー定義スクリプトＤ４３ｂを含む。

【００６４】また、これらのスクリプトＤ４３ａおよびＤ４３ｂを相互に識別するため、各スクリプトＤ４３ａおよびＤ４３ｂには異なる識別子が付加されている。図５に示す例では、識別子engines および識別子flowがフロー定義スクリプトＤ４３ｂを示し、識別子doがオブジェクト定義スクリプトＤ４３ａを示す。

【００６５】識別子engines はワークフローエンジンサブクラス名を指定する。ここでは、各処理者３１ａおよび３１ｂ毎に異なるワークフローエンジンサブクラスコードが生成されているため、ワークフローエンジンサブクラス名によって各アクティビティを実行する処理者３１ａおよび３１ｂを定義している。例えば、図５に示すように、処理者３１ａに対してはワークフローエンジンサブクラスＦ２７ａ（loanTeller）を定義し、処理者３１ｂに対してはワークフローエンジンサブクラスＦ２７ｂ（loanDecision）を定義している。

【００６６】識別子flowは、ワークフローエンジンサブクラス名を順次に指定することにより、アクティビティの処理順序を定義する。すなわち図３では、ワークフローエンジンサブクラスＦ２７ａ（loanTeller）およびＦ２７ｂ（loanDecision）の順に、アクティビティが実行される。

【００６７】識別子doは、各アクティビティにおける処理内容を定義する。すなわち図５に示すように、ワークフローエンジンサブクラスＦ２７ａ（loanTeller）ではメソッドstart という処理内容が定義され、ワークフローエンジンサブクラスＦ２７ｂ（loanDecision）ではメソッドdecideLoanという処理内容が定義される。

【００６８】図３に示すように、このようなプロセス定義スクリプトＤ４３は、ホスト計算機ｈ１のＣＰＵｃ１で動作するスクリプト識別手段Ｆ４５に読み込まれる。このとき、スクリプト識別手段Ｆ４５は、プロセス定義スクリプトＤ４３に含まれる識別子に基づいて、プロセス定義スクリプトＤ４３をオブジェクト定義スクリプトＤ４３ａおよびフロー定義スクリプトＤ４３ｂに分離する。その後、スクリプト識別手段Ｆ４５は、ホスト計算機ｈ１に分離したオブジェクト定義スクリプトＤ４３ａおよびフロー定義スクリプトＤ４３ｂを記憶装置ｍ１に格納する。なお、スクリプト識別手段Ｆ４５は、図４に示すホスト計算機ｈ１の記憶装置ｍ１に格納されているスクリプト識別プログラム（図示せず）をＣＰＵｃ１が読み出してかつ実行することにより、機能する仮想的な機能単位である。

【００６９】なお、プログラム格納部２１は、具体的に言うと、次のような基本機能提供プログラムＤ２２を有する。すなわち、プログラム格納部２１は、例えば、

①新規にワークフロー処理を開始させる機能を実現する基本機能提供プログラムＤ２２と、
②実行中のワークフロー処理の進行状況を確認する機能を実現する基本機能提供プログラムＤ２２と、
③ワークフロー処理を中止する機能を実現する基本機能提供プログラムＤ２２と、
④ワークフロー処理における全てのアクティビティの終了を確認する機能を実現する基本機能提供プログラムＤ２２と、
⑤中止または終了したワークフロー処理に関する情報を破棄する機能を実現する基本機能提供プログラムＤ２２と、
⑥処理者毎に設けられたワークリストにワークアイテムを付加する機能を実現する基本機能提供プログラムＤ２２と、
⑦ワークリストにおけるワークアイテムの削除によってアクティビティの終了を認識する機能を実現する基本機能提供プログラムＤ２２とを有する。

【００７０】なお、図４に示すように、ここでは階層構造を有するワークフローエンジンのクラスがホスト計算機ｈ１、ｈ２およびｈ３に分散して存在しているため、プログラム格納部２１は、⑧ワークフローエンジンサブクラスが他のワークフローエンジンと相互に通信するための通信機能を実現する基本機能提供プログラムＤ２２を有するのが好ましい。また、プログラム格納部２１は、⑨人間による処理を必要としないアクティビティについて、起動アプリケーションによってアクティビティを自動処理する基本機能提供プログラムＤ２２を有しても良い。また、図４に示すように、プログラム格納部２１は、例えばホスト計算機ｈ１の記憶装置ｍ１のデータ領域として設けられる。

【００７１】また、Ｊａｖａ言語によるＯＯＰを行うには、プログラム格納部２１が、ワークフローエンジンのスーパークラス等のワークフロー処理に利用できるクラスを定義したクラスライブラリ５２を有するのが望ましい。図３では、クラスライブラリＤ５２はワークフローエンジンスーパークラスコードＤ２７ｓ（DWorkflowEngine）およびその他のスーパークラスコードＤ２７ｔ（LoanApp2）を含む。

【００７２】ワークフローエンジンコード生成手段Ｆ１９は、記憶装置ｍ１のプログラム格納部２１から上述のような①～⑨の基本機能提供プログラムＤ２２、記憶装置ｍ１に蓄積されたオブジェクト定義スクリプトＤ４３ａおよびフロー定義スクリプトＤ４３ｂを読み出したのち、読み出された①～⑨の基本機能提供プログラムＤ２２を組み合わせて、ワークフローエンジンサブクラスＦ２７ａ（LoanTeller）およびＦ２７ｂ（LoanDecision）として機能するワークフローエンジンサブクラスコードＤ２７ａ（LoanTeller）およびＤ２７ｂ（LoanDecision）を生成する。

【００７３】このとき、ワークフローエンジンコード生成手段Ｆ１９は、ワークフロー処理に関するオブジェクト定義スクリプトＤ４３ａおよびフロー定義スクリプトＤ４３ｂを解釈することにより、当該ワークフロー処理に必要な基本機能のみを有するワークフローエンジンサブクラスコードＤ２７ａ（LoanTeller）およびＤ２７ｂ（LoanDecision）を生成する。すなわち、ワークフローエンジンコード生成手段Ｆ１９は、オブジェクト定義スクリプトＤ４３ａおよびフロー定義スクリプトＤ４３ｂで定義されたワークフロー処理に必要な基本機能のみを読み込みかつ不必要な機能を読み込まない。例えば、スクリプト識別手段Ｆ４５から読み出されたオブジェクト定義スクリプトＤ４３ａおよびプログラム格納部２１から読み出された①～⑨の基本機能の幾つかを組み合わせて、ワークフローエンジンサブクラスコードＤ２７ａ（LoanTeller）およびＤ２７ｂ（LoanDecision）が生成される。そのため、このワークフローエンジンサブクラスコードＤ２７ａ（loanTeller）およびＤ２７ｂ（LoanDecision）は、必要最低限の構成のみを有し、従って占有メモリが小さくなる。具体的に言うと、例えば、従来のワークフローエンジンでは任意のステップ数のアクティビティが処理できなければならないが、実施の形態のワークフローエンジンは、２つのアクティビティが処理できればよいため、余分なクラスやメソッドの定義を省略したコードとなる。

【００７４】オブジェクト定義コード生成手段Ｆ１５は、記憶装置ｍ１のプログラム格納部２１から上述のような基本機能提供プログラムＤ２２と、記憶装置ｍ１に蓄積されたオブジェクト定義スクリプトＤ４３ａとを読み出したのち、読み出された基本機能提供プログラムＤ２２およびオブジェクト定義スクリプトＤ４３ａを組み合わせてオブジェクト定義コードＤ４７ａを生成する。

【００７５】フロー定義コード生成手段Ｆ１７は、記憶装置ｍ１のプログラム格納部２１から上述のような基本機能提供プログラムＤ２２と、記憶装置ｍ１に蓄積されたフロー定義スクリプトＤ４３ｂとを読み出したのち、読み出された基本機能提供プログラムＤ２２およびフロー定義スクリプトＤ４３ｂを組み合わせてフロー定義コードＤ４７ｂを生成する。

【００７６】図６および図７は、オブジェクト定義コード生成手段Ｆ１５およびフロー定義コード生成手段Ｆ１７によってＪａｖａ言語によるソースプログラムコードとして生成されたオブジェクト定義コードＤ４７ａおよびフロー定義コードＤ４７ｂの一部を示した図である。

【００７７】図６に示すソースプログラムコードのうち１～２行目のコードは、ワークフローエンジンのスーパークラスコードＤ２７ｓ（DWorkflowEngine）に対するサブクラスコードＤ２７ａ（loanDecision）を定義する。３～７行目のコードは、ワークフローエンジンサブクラスコードＤ２７ｂ（loanDecision）について例外処

理を宣言している。８行目～１１行目のコードは、ワークフローエンジンサブクラスコードＤ２７ｂ（loanDecision）における処理内容を示すメソッドdecideloanを定義する。また、１２～２４行目のコードは、ワークフローエンジンサブクラスコードＤ２７ｂ（loanDecision）を開始したとき最初に実行されるメソッドが定義される。

【００７８】図７に示すソースプログラムコードのうち１～２行目のコードによって、ワークフロー処理に必要なその他のスーパークラスコードＤ２７ｔ（LoanApp2）のサブクラスLoanFlow2 が定義される。３～７行目のコードは、その他のスーパークラスコードＤ２７ｔ（LoanFlow2 ）の初期化および名前の定義を行う。８～２６行目のコードは、loanFlow2 に用いるワークフローエンジンサブクラスコードＤ２７ａ（loanTeller）およびＤ２７ｂ（loanDecision）を登録する。２７～３１行目のコードは、ワークフローエンジンサブクラスコードＤ２７ａ（loanTeller）およびＤ２７ｂ（loanDecision）の名前のテーブルを作成する。３２～３４行目のコードは、ワークフロー処理の名前のテーブルを作成する。３５～３７行目のコードは、ワークフロー処理のタイムリミットテーブルを作成する。３８～４０行目のコードは、ワークフロー処理の終了を判定するテーブルを作成する。

【００７９】また、実際にワークフロー処理を行うとき、図３に示すワークフローエンジンサブクラスＦ２７ａ（loanTeller）およびＦ２７ｂ（LoanDecision）が駆動して、オブジェクト定義コードＤ４７ａおよびフロー定義コードＤ４７ｂのクラスやメソッドの参照を行うことにより、ワークフロー処理を制御する。

【００８０】また、実際にワークフロー処理を行うとき、オブジェクト定義コードＤ４７ａおよびフロー定義コードＤ４７ｂには、プロセスに関する定義が記述されていなければならない。これらの定義コードＤ４７ａおよびＤ４７ｂは、一つのステップで生成しても良いし、或いは複数ステップに分けて生成しても良い。例えば、コードジェネレータＦ２８では基本的な定義コードのみを生成し、基本的な定義コードに具体的な定義コードを追加することにより、定義コードを完成させても良い。言い換えれば、複数ステップに分けてオブジェクト定義コードＤ４７ａやフロー定義コードＤ４７ｂを生成する場合、先ずオブジェクト定義基本コードとして各アクティビティに共通の処理内容を定義するコードを生成し、および、オブジェクト定義拡張コードとして個々のアクティビティに固有の処理内容を定義するコードをオブジェクト定義基本コードに追加することにより、オブジェクト定義コードを生成しても良い。具体的に言うと、図６のオブジェクト定義コードＤ４７ａは、８～９行目のブロックにオブジェクト定義拡張コードとして、個々のアクティビティに固有のメソッドが追加されると、オブジェクト定義コードとして完成する。このように、オブジェクト定義コード生成手段Ｆ１５がオブジェクト定義基本コードを生成したのち、オブジェクト定義基本コードにオブジェクト定義拡張コードを付加するのが好ましい。それにより、個々のオブジェクト定義コードが簡潔となる。

【００８１】なお、説明の都合上、仮想的な各機能単位である識別手段、ワークフローエンジンコード生成手段、オブジェクト定義コード生成手段、フロー定義コード生成手段を特定の同一ホスト計算機のＣＰＵに存在させたが、これら各機能単位は任意のホスト計算機のＣＰＵに存在できる。また、プログラム格納部２１、オブジェクト定義コードＤ４７ａ、フロー定義コードＤ４７ｂも任意のホスト計算機の記憶装置に存在して良い。

【００８２】以上のような手順で、プロセス定義スクリプトでワークフロー処理のプロセスを定義することにより、ワークフローエンジンサブクラスコードＤ２７ａ（LoanTeller）およびＤ２７ｂ（LoanDecision）と、オブジェクト定義コードＤ４７ａと、フロー定義コードＤ４７ｂとが生成できる。ただし、ワークフロー処理の開始の際には、ホスト計算機ｈ１で生成されたワークフローエンジンサブクラスコードＤ２７ａ（LoanTeller）およびＤ２７ｂ（LoanDecision）を、それぞれホスト計算機ｈ２およびホスト計算機ｈ３に分散させる。

【００８３】従来のワークフロー管理システムと同様に、図３に示す各ワークフローエンジンサブクラスＦ２７ａ(loanTeller)およびＦ２７ｂ（LoanDecision）と同一ホスト計算機には、各々ワークリストＤ５３ａおよびＤ５３ｂと、ワークリストハンドラＦ５５ａおよびＦ５５ｂとが設けられている。ワークリストＤ５３ａおよびＤ５３ｂは、個々の処理者３１ａおよび３１ｂ毎に設けられる。

【００８４】各ホスト計算機ｈ２およびｈ３では次のような処理が行われる。すなわち、先ず、各処理者３１ａは、ワークフローエンジンサブクラスＦ２７ａ(loanTeller)によってワークリストＤ５３ａに示されたワークアイテムを参照する。それにより、処理者３１ａは処理すべきアクティビティを認識する。そして、処理者３１ａはクライアントアプリケーションＤ５７等を用いてアクティビティを実行する。アクティビティに含まれる作業が終了すると、ワークリストハンドラＦ５５ａはワークリストＤ５３ａからワークアイテムを削除する。それにより、ワークフローエンジンサブクラスＦ２７ａ（loanTeller）は、アクティビティの終了を検出する。その後、ワークフローエンジンＦ２７ａ（loanTeller）は、アクティビティで処理されたデータを次のワークフローエンジンＦ２７ｂ（loanDecision）に送付する。

【００８５】そして、各処理者３１ｂは、ワークフローエンジンサブクラスＦ２７ｂ（loanDecision）によってワークリストＤ５３ｂに示されたワークアイテムを参照する。処理者３１ｂは処理すべきアクティビティを認識

する。そして、処理者３１ｂはクライアントアプリケーションＤ５７等を用いてアクティビティを実行する。アクティビティが終了すると、処理者３１ｂはワークリストハンドラＦ５５ｂを操作してワークリストＤ５３ｂからワークアイテムを削除する。それにより、ワークフローエンジンサブクラスＦ２７ｂ（loanDecision）は、アクティビティの終了を検出する。同時に、ワークフローエンジンＦ２７ｂ（loanDecision）は、ワークフロー処理の全てのアクティビティの終了を検出して、例えば処理者３１ａ等にワークフロー処理の終了および結果を通知する。

【００８６】続いて、上述のワークフロー管理システムを一適用例としてワークフロー処理のプロセスを変更する方法につき説明する。

【００８７】実施の形態のワークフロー管理システムのプロセスの変更は、オブジェクト定義コードＤ２５ａ（Ｄ４７ａ）或いはフロー定義コードＤ２５ｂ（Ｄ４７ｂ）を書き換えて行う。したがって、従来技術のワークフロー管理システムのようにプロセス定義データそのものを書き換える必要がない。すなわち、実施の形態のワークフロー管理システムによれば、各アクティビティの処理内容のみを変更したい場合にはオブジェクト定義コードＤ２５ａのみを修正し、アクティビティの順序のみを変更したい場合にはフロー定義コードＤ２５ｂのみを修正すればよい。

【００８８】具体的に言うと、例えば図７のフロー定義コードに従うワークフロー処理において、処理者を変更したい場合には、フロー定義コードに記述されたワークフローエンジンサブクラス（loanDecision）を別のワークフローエンジンサブクラスに名称の変更をすればよい。また、例えば図６のオブジェクト定義コードに従うワークフロー処理において、ワークフローエンジンサブクラス（loanDecision）の処理内容を変更したい場合には、オブジェクト定義コードの処理内容を示すメソッドdecideLoanを修正すればよい。このように、オブジェクト定義コード或いはフロー定義コードの一方の修正が他方の修正に影響しないため、プロセスは容易に変更できる。

【００８９】オブジェクト定義コード或いはフロー定義コードを修正するには、前述のワークフロー管理システムが、オブジェクト定義コードおよび前述のフロー定義コードを高水準言語のコードの状態で保存する定義コード保存手段（図示していない。）を更に具えるのが望ましい。

【００９０】図６および図７に示すオブジェクト定義コードおよびフロー定義コードは、ワークフロー処理を開始する前に生成される。これらの定義コードはこの実施の形態ではソースプログラムである。これらの定義コードはコンパイルされることにより、動作可能なオブジェクトプログラムとなる。例えば、Ｊａｖａ言語ソースプログラムをコンパイルすると、ソースプログラムはプラットフォームに依存せずに動作するバイトコードに変換されて実行型プログラムとなる。コンパイルしたバイトコードデータそのものを書き換えるのは困難であるから、コンパイルする前の状態すなわち例えばＪａｖａ言語のソースプログラムコードを保存しておく。それにより、各アクティビティにおける処理内容或いは処理順序を変更する際、Ｊａｖａ言語のソースプログラムコードの一部を修正することにより、新たなオブジェクト定義コード或いはフロー定義コードが作成できる。新たな定義コードを、コンパイルしてバイトコードに変換したのち修正前のバイトコードデータと置換すれば、プロセスに関する定義が修正できる。

【００９１】（第２の実施の形態）以下の第２の実施の形態では、第１の実施の形態のワークフロー管理システムに宛名変換手段を設けた形態につき説明する。ただし、ここでは特に階層構造を有するワークフローエンジンを設けている形態につき説明する。第１の実施の形態にて説明したワークフロー管理システムでは、アクティビティの処理順序はフロー定義コードに静的に定義される。すなわち、アクティビティの処理者を示すコードには具体的な宛先が示されており、人事異動等による宛先の変更すなわちアクティビティの順序の変更があった場合、フロー定義コードを書き換える必要がある。

【００９２】ところで、文献Ｉ等に開示されている宛名変換手段を用いると、動的に処理順序を定義できるため、プロセス定義データを修正する必要が無くなることが知られている。しかしながら、文献Ｉに開示されるワークフロー管理システムの場合、ワークフローエンジンはサーバで、また、宛名変換手段はワークフローエンジンと同一のサーバで動作する。そのため、宛名変換手段を含むサーバの負担が大きくなる。また、実際にサーバがダウンしたときには、ワークフロー処理が中断されてしまうという問題があった。

【００９３】図８は、第２の実施の形態のワークフロー管理システムを動作させるワークフロー管理システムの各機能単位を模式的かつ概略的に示すブロック図である。ただし、図８では図４と同一の機能単位については示しておらず、図４と異なる点のみすなわちワークフローエンジンサブクラスＦ２７ａ、Ｆ２７ｂおよび宛名変換手段Ｆ６７が示される。

【００９４】図８に示すように、第２の実施の形態のワークフロー管理システムは、宛名変換手段Ｆ６３を有する。ここで、ホスト計算機ｈ４のＣＰＵｃ４が記憶装置ｍ４の宛名変換プログラムＤ６３を読み出しかつ実行することにより、宛名変換手段Ｆ６３は仮想的な機能手段としてＣＰＵｃ４で動作する。

【００９５】図８に示すワークフローエンジンは、ＯＯＰによってプログラミングされており、スーパークラスから複数のサブクラスＦ２７ａおよびＦ２７ｂに機能を

継承するクラス構造を有する。そして、個々のワークフローエンジンサブクラスＦ２７ａおよびＦ２７ｂは、各処理者毎に分散して存在するため、互いにアクセスするための各通信機能Ｆ６７ａおよびＦ６７ｂを有する。

【００９６】そして、この第２の実施の形態の宛名変換手段Ｆ６３は、特に、通信機能Ｆ６７ｔを備える点を特徴とする。よって、宛名変換手段Ｆ６３と、複数のワークフローエンジンサブクラスＦ２７ａおよびＦ２７ｂとは通信によって互いにアクセスできる。

【００９７】宛名変換手段Ｆ６３は、周知のごとく、課長や部長など組織内の役職等を示す一般的宛先名を、処理者の名前等の具体的宛先名に変換する機能モジュールである。また、この宛名変換手段Ｆ６３は、従来構成と同様に組織／役割データＤ６５を参照する。例えば備品購入の決裁のワークフロー処理を行う場合、決裁ルートは役職で決定されることが多いから、予め宛先名を役職で指定する。そのため、決裁ルートに沿う役職にある処理者が人事異動等によって他の処理者に変わっても動的に宛先が判断されるため、決裁ルートすなわち宛先名を変更する必要が無い。

【００９８】ここで図８を参照して、宛名変換手段６３によって一般的宛先名から具体的宛先名を判断する流れについて説明する。

【００９９】ワークフローエンジンサブクラスＦ２７ａは、フロー定義コード中に定義されたアクティビティの処理者が一般的な宛先名で記述されている場合、先ず、そのワークフロー処理を開始させたワークフローエンジンサブクラス名すなわち処理者を調べる。次に、ワークフローエンジンサブクラスＦ２７ａは、各通信機能Ｆ６７ａおよびＦ６７ｔを介して宛名変換手段Ｆ６３とネットワーク通信することにより、一般的宛先名および開始者名を宛名変換手段Ｆ６３に送信する。

【０１００】このとき、宛名変換手段６３は、実行中のワークフロー処理の開始者名から開始者の組織内の役割（所属部署や役職等）を認識すると共に、この開始者の役割および一般的宛先名から次のアクティビティを担当する処理者名を調べる。そして、次の処理者名からワークフローエンジンサブクラス名を得て、次のアクティビティを処理すべきワークフローエンジンサブクラス名を、ワークフローエンジンサブクラスＦ２７ａに送信する。そして、この後、次のアクティビティを処理べきワークフローエンジンサブクラスＦ２７ｂが起動する。

【０１０１】以上のように、第２の実施の形態では、図８に示すようにワークフローエンジンが階層構造をなす複数のワークフローエンジンサブクラスＦ２７ａおよびＦ２７ｂとして異なるホスト計算機ｈ３およびｈ４に分散しており、および、宛名変換手段Ｆ６３が通信機能を有する。すなわち、ホスト計算機ｈ２ではワークフローエンジンサブクラスＦ２７ａが動作し、ホスト計算機ｈ３ではワークフローエンジンサブクラスＦ２７ｂが動作し、および、ホスト計算機ｈ４では宛名変換手段Ｆ６３が動作する。このように、宛名変換手段Ｆ６３と、ワークフローエンジンサブクラスＦ２７ａおよびＦ２７ｂとが異なるホスト計算機に存在しても、宛名変換手段Ｆ６３が通信機能Ｆ６７ｔを有するため、宛名変換手段Ｆ６３はワークフローエンジンサブクラスＦ２７ａおよびＦ２７ｂと互いに通信できる。したがって、宛名変換手段Ｆ６３を任意のホスト計算機に移動させることができる。すなわち、従来のように宛名変換手段Ｆ６３をワークフローエンジンと同一ホストに設ける必要がない。

【０１０２】そのため、宛名変換手段を含むホスト計算機の負担を小さくしシステムダウンの可能性を低減させることができる。また、宛名変換手段を含むホスト計算機がシステムダウンしたときでも、別のホスト計算機に宛名変換手段を移動することによって、ワークフロー処理を続行できる。

【０１０３】

【発明の効果】上述した説明から明らかなように、この発明のワークフロー管理システムによれば、ワークフロー処理のプロセスを定義をする際、起票者が入力したプロセス定義情報は、識別手段によってフロー定義に関する部分およびオブジェクト定義に関する部分に分離されたのち、フロー定義コード生成手段およびオブジェクト定義コード生成手段によってコード化される。そのため、ワークフロー処理を行うとき、各アクティビティにおける処理内容およびアクティビティの順序を変更する場合であっても、プロセス定義データそのものを修正する必要がない。言い換えれば、各アクティビティにおける処理内容のみを変更したい場合にはオブジェクト定義コードのみを修正すれば良く、アクティビティの順序（アクティビティを行う処理者の変更）を変更したい場合にはフロー定義コードのみを修正すればよい。したがって、プロセスを能率的にかつ容易に変更することができる。

【０１０４】また、この発明のコードジェネレータによれば、識別手段が、プロセス定義情報をオブジェクト定義情報およびフロー定義情報に分離させたのち、オブジェクト定義情報からオブジェクト定義コードを生成しかつフロー定義情報からフロー定義コードを生成する。そのため、各工程における処理内容のみを変更したい場合にはオブジェクト定義コードを修正し、工程の処理順序のみを変更したい場合にはフロー定義コードを修正すればよい。よって、プロセスを能率的にかつ容易に変更することができる。

【０１０５】また、このワークフロー処理のプロセスの変更方法の発明によれば、プロセス定義情報から、オブジェクト定義コードおよびフロー定義コードを生成するワークフロー処理のプロセスを変更するのに、各工程の処理内容のみを変更する場合にはオブジェクト定義コードのみを修正し、工程の処理順序のみを変更する場合に

はフロー定義コードのみを修正する。したがって、このプロセスの変更方法によれば、各工程における処理内容のみを変更したい場合若しくは工程の処理順序のみを変更したい場合のいずれにも、互いに独立に存在するオブジェクト定義コード或いはフロー定義コードの一方を修正することにより対応できる。よって、プロセスを能率的にかつ容易に変更できる。

【図面の簡単な説明】

【図１】第１の実施の形態のワークフロー管理システムの機能単位の相互関係を概略的に示すブロック図である。

【図２】図１の各機能単位を１台のホスト計算機で動作させた場合を例として、各機能単位によるハードウェア資源の利用形態を概略的に示す図である。

【図３】図１のワークフロー管理システムの具体的な機能構成を概略的に示した図である。

【図４】図３のワークフロー管理システムの各機能単位を動作させるネットワークシステムの一例を示す図である。

【図５】具体的なワークフロー処理に対するプロセス定義スクリプトの一例である。

【図６】ソースプログラムコードとして生成されたオブジェクト定義コードの一部を示した図である。

【図７】ソースプログラムコードとして生成されたフロー定義コードの一部を示した図である。

【図８】第２の実施の形態のワークフロー管理システムを動作させるネットワークシステムの各機能単位を模式的かつ概略的に示すブロック図である。

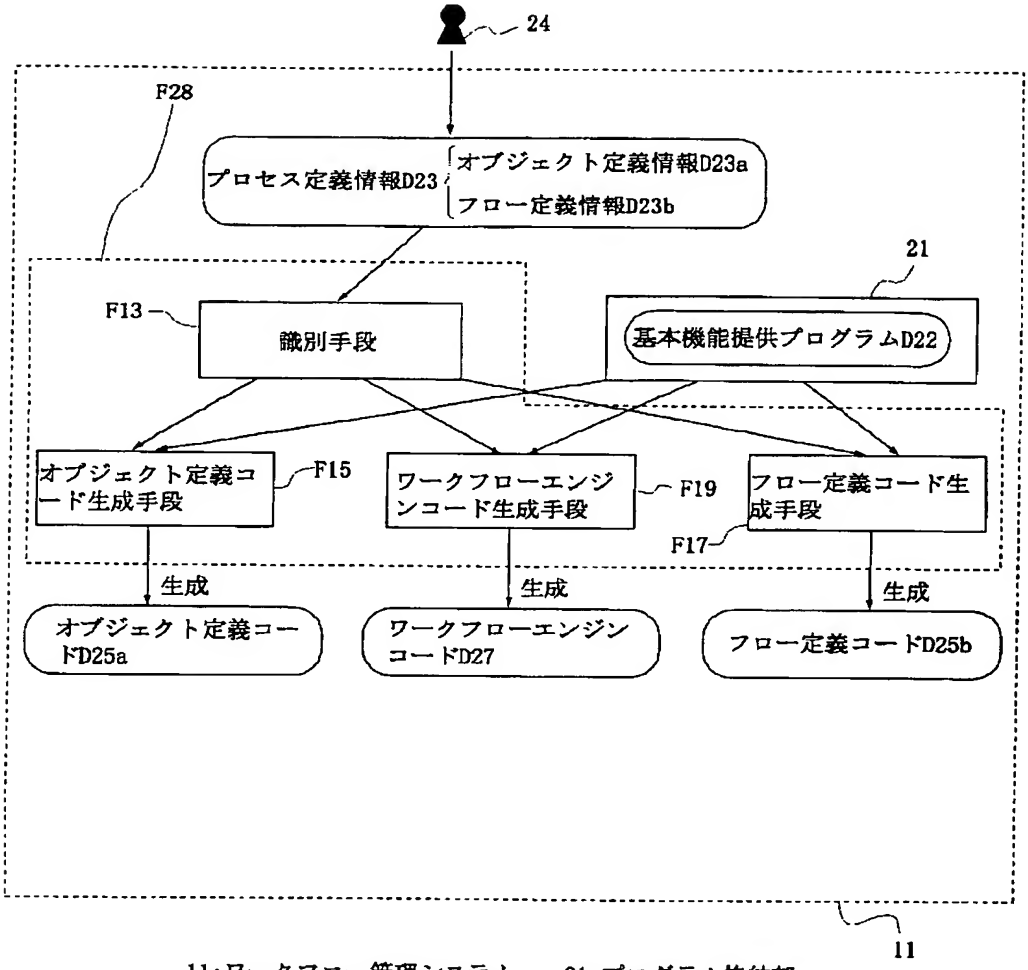
【図９】従来のワークフロー管理システムの構成を示すブロック図である。

【符号の説明】

１１、４１：ワークフロー管理システム
Ｆ１３：識別手段
Ｄ１３：識別プログラム

Ｆ１５：オブジェクト定義コード生成手段
Ｄ１５：オブジェクト定義コード生成プログラム
Ｆ１７：フロー定義コード生成手段
Ｄ１７：フロー定義コード生成プログラム
Ｆ１９：ワークフローエンジンコード生成手段
Ｄ１９：ワークフローエンジンコード生成プログラム
２１：プログラム格納部
Ｄ２２：基本機能提供プログラム
Ｄ２３：プロセス定義情報
Ｄ２３ａ：オブジェクト定義情報
Ｄ２３ｂ：フロー定義情報
２４：起票者
Ｄ２５ａ、Ｄ４７ａ：オブジェクト定義コード
Ｄ２５ｂ、Ｄ４７ｂ：フロー定義コード
Ｄ２７：ワークフローエンジンコード
Ｆ２７ａ、Ｆ２７ｂ：ワークフローエンジンサブクラス
Ｄ２７ａ、Ｄ２７ｂ：ワークフローエンジンサブクラスコード
Ｄ２７ｓ：ワークフローエンジンスーパークラスコード
Ｄ２７ｔ：その他のスーパークラスコード
Ｆ２８：コードジェネレータ
３１ａ、３１ｂ：処理者
Ｄ４３：プロセス定義スクリプト
Ｄ４３ａ：オブジェクト定義スクリプト
Ｄ４３ｂ：フロー定義スクリプト
Ｆ４５：スクリプト識別手段
Ｄ５２：クラスライブラリ
Ｄ５３ａ、Ｄ５３ｂ：ワークリスト
Ｆ５５ａ、Ｆ５５ｂ：ワークリストハンドラ
Ｄ５７：クライアントアプリケーション
Ｆ６３：宛名変換手段
Ｄ６３：宛名変換プログラム
Ｄ６５：組織／役割データ
Ｆ６７ａ、Ｆ６７ｂ、Ｆ６７ｔ：通信機能

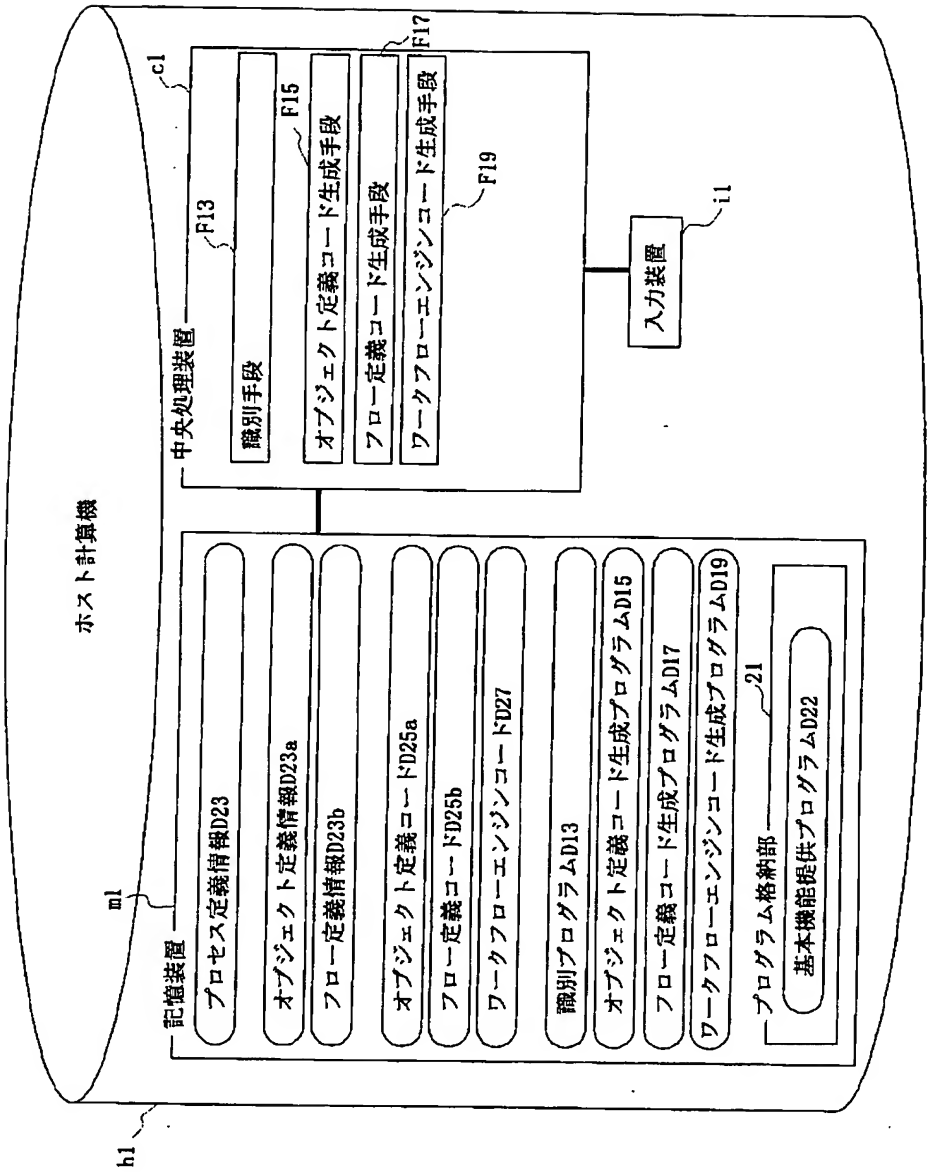
【図１】



11:ワークフロー管理システム　　21:プログラム格納部
24:起票者　　F28:コードジェネレータ

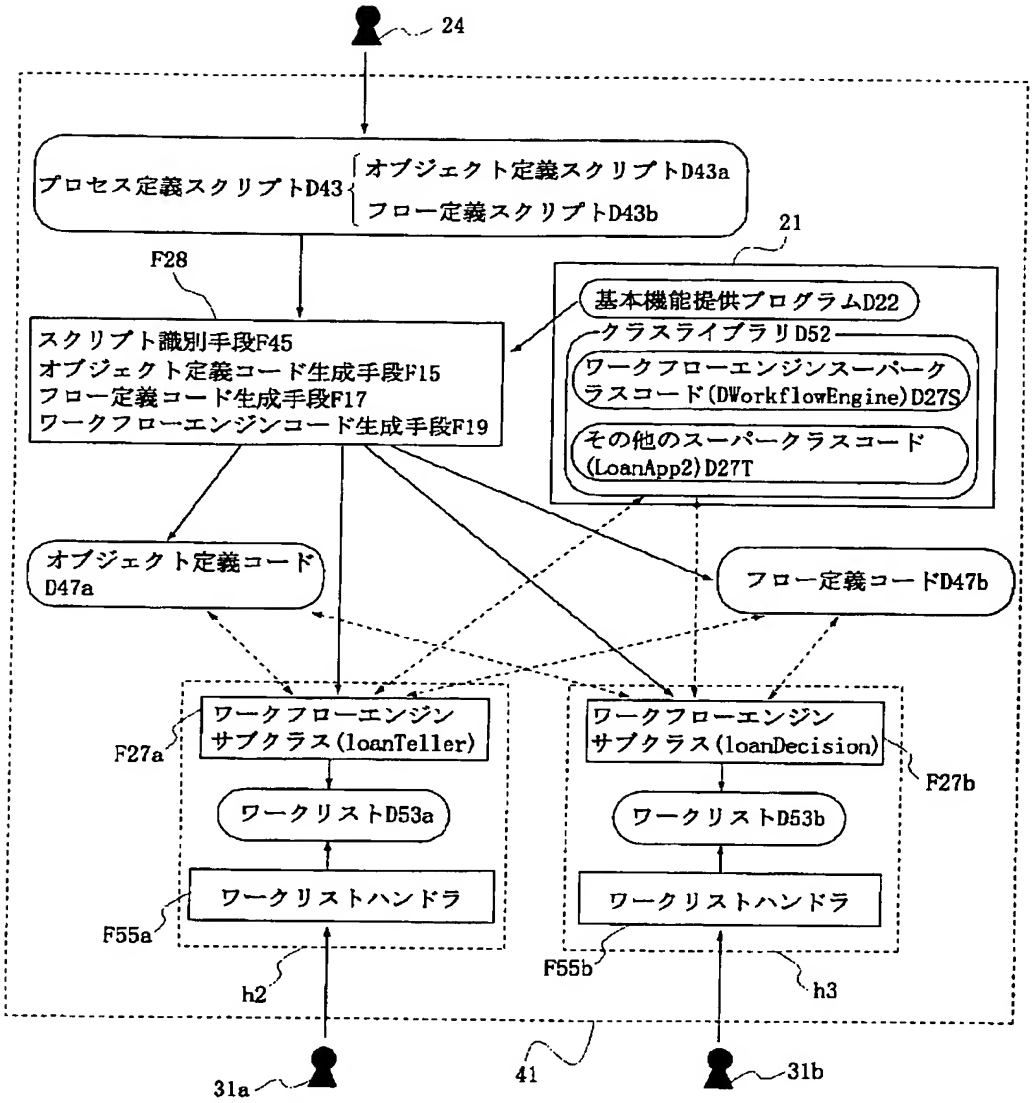
第１の実施の形態のワークフロー管理システム

【図２】



ハードウェア資源の利用の形態の説明に供する図

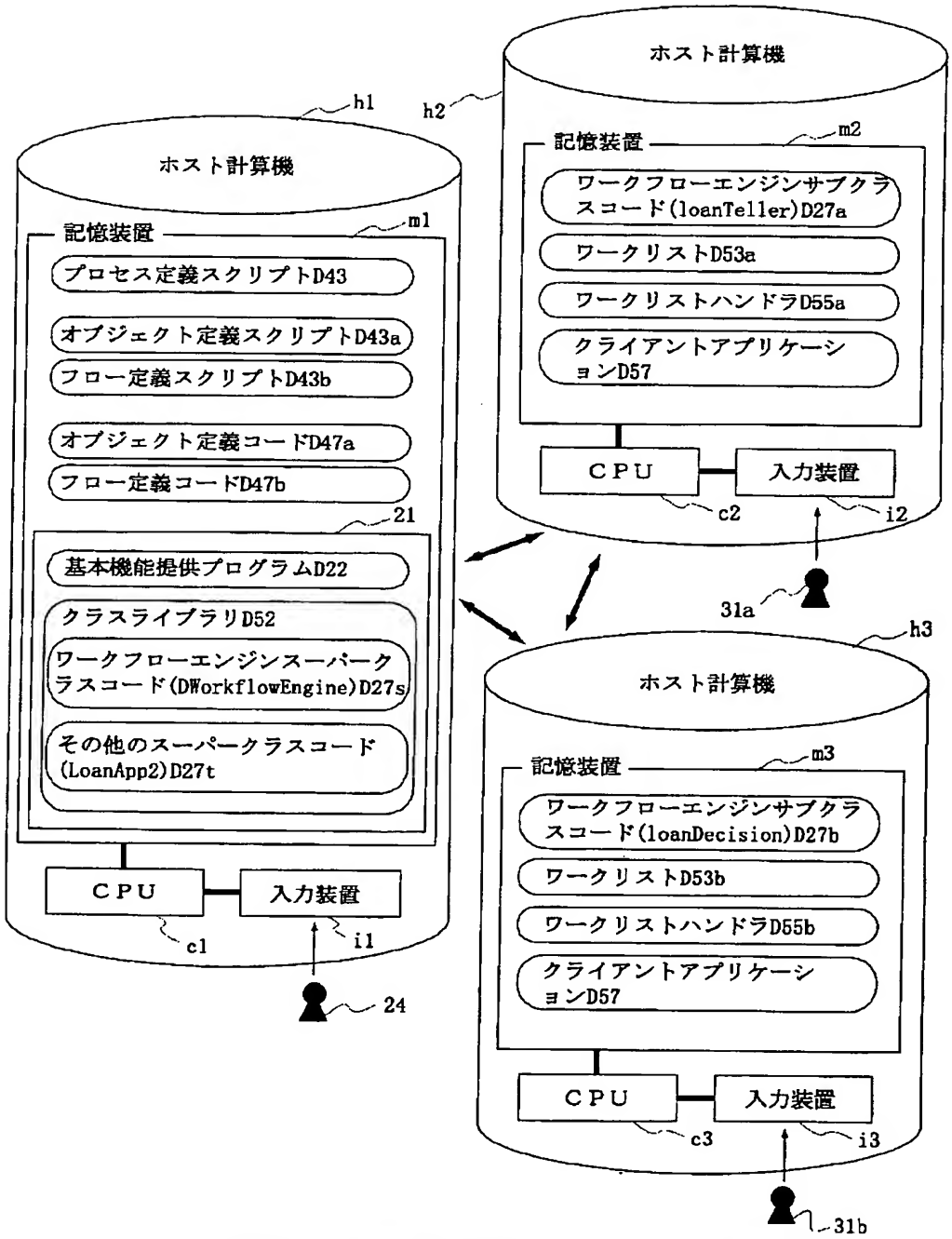
【図３】



h2, h3: ホスト計算機　　31a, 31b: 処理者　　41: ワークフロー管理システム

第1の実施の形態のワークフロー管理システムの具体的な機能構成

【図４】



ワークフロー管理システムを動作させるネットワークシステムの構成例

【図５】

```
aspect Flow relates LoanApp2
   engines {
      loanTeller ―"loanTeller";
      loanDecision ―"loanDecision";
   }
   flow {                                    } D43b
      loanTeller;
      loanDecision;
      loanTeller;
   }
   do (loanTeller) {                         }
      %E.start(%W);                          } D43a
   }
   do (loanDecision) {                       }
      %E.decideLoan(%W);
   }
}
```

プロセス定義スクリプトの一例

【図６】

```
1:  public class LoanDecison extends DWorkflowEngine
2:  {
3:     public LoanDecision(String name)
4:        throws java.rmi.RemoteException
5:     {
6:        super(name);
7:     }
8:     public void decideLoan(MessageObject[] mos)
9:     {
10:       System.out.println("invoke decideLoan()");
11:    }
12:    public static void main(String arg[])
13:       throws Exception
14:    {
15:       try {
16:           System.out.println("LoanDecision boot");
17:           LoanDecision de = new LoanDecision("LoanDecision");
18:           bind(de);
19:           de.act();
20:       } catch(Exception e) {
21:           e.printStackTrace();
22:       }
23:    }
24: }
```

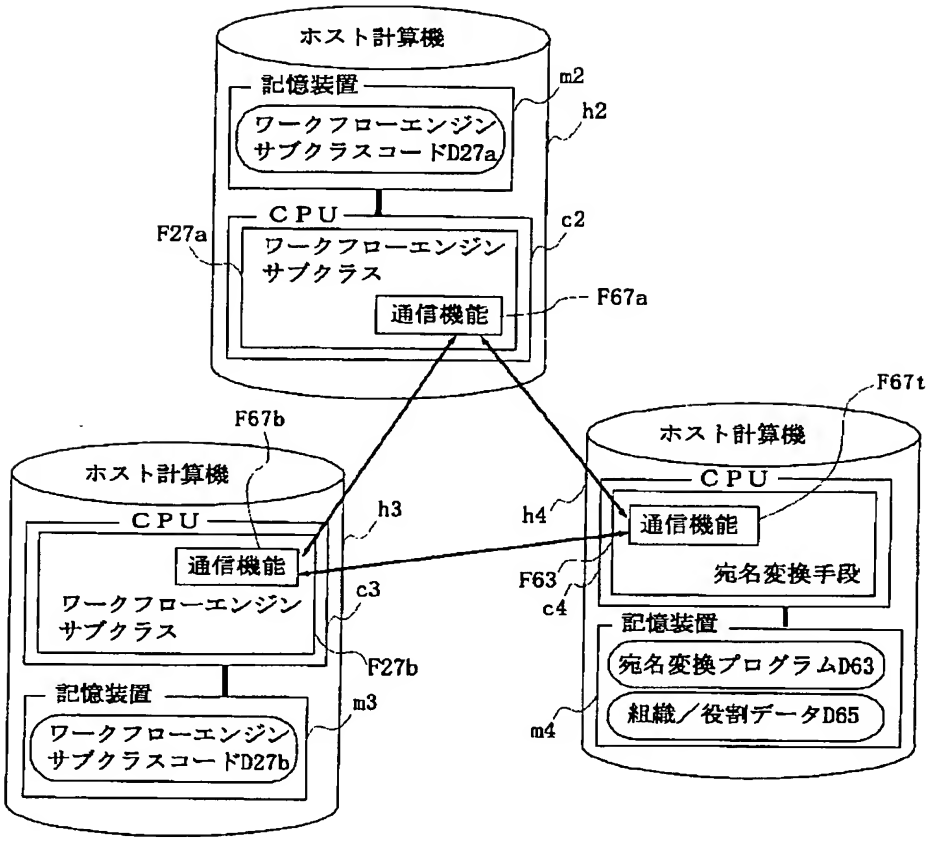
Ｊａｖａソースプログラムコードによるオブジェクト定義コードの一部

【図７】

```
1:  public class LoanFlow2 extends LoanApp2
2:  {
3:     public LoanFlow2(WorkItems workitems)
4:     {
5:        super(workitems);
6:        flowName = "LoanFlow2";
7:     }
8:     public void makeFlowTree()
9:     {
10:       DEItem deItem = null;
11:       NFItem nfItem = null;
12:       BasicFlowElement flow0 = null;
13:
14:       // DE:loanTeller
15:       deItem = new DEItem();
16:       deItem.name = "loanTeller";
17:       link(flow0, deItem);
18:       deVSTreeTable.put(deItem.name, deItem);
19:
20:       // DE:loanDecision
21:       deItem = new DEItem();
22:       deItem.name = "loanDecision";
23:       link(flow0, deItem);
24:       deVSTreetable.put(deItem.name, deItem);
25:
26:    }
27:    public void makeDENameTable()
28:    {
29:       deNameTable.put("loanDecision","loanDecision");
30:       deNameTable.put("loanTeller","loanTeller");
31:    }
32:    public void makeFlowNameTable()
33:    {
34:    }
35:    public void makeTimeLimitTable()
36:    {
37:    }
38:    public void makeAcceptTable()
39:    {
40:    }
41: }
```

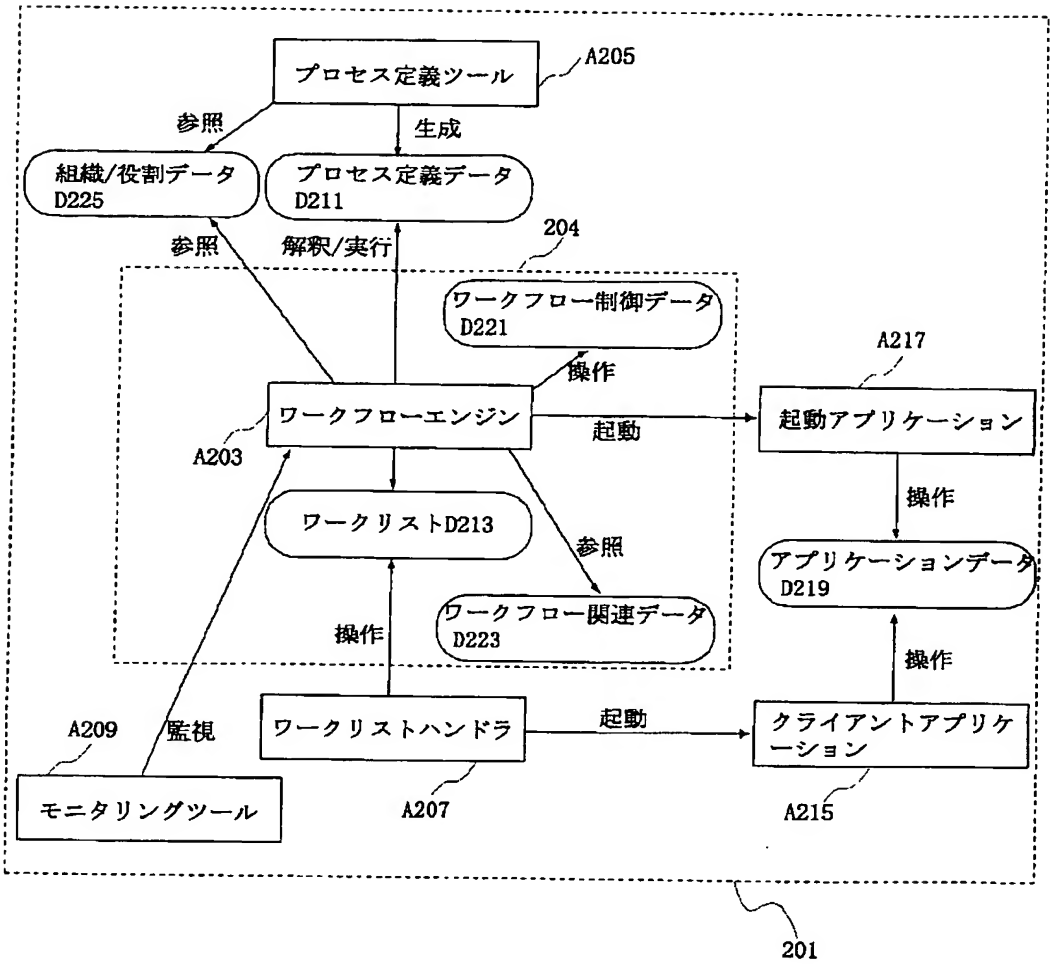
Ｊａｖａソースプログラムコードによるフロー定義コードの一部

【図８】



第２の実施の形態のワークフロー管理システムを動作させるネットワークシステムの構成例

【図９】



従来のワークフロー管理システム